CONCEPT OF MOV AX,CS and MOV DS,AX

can someone please explain the functions of the three instructions.....

```
>       ORG 1000H
>       MOV AX,CS
>       MOV DS,AX
```

I know what is code , data , extra segments in theory.

But,

1.How they are implementing in these program?

2.Why they move the entire segment into another?**(MOV AX,CS;MOV DS,AX;**)

(excuse me if question is absurd) what is the **real concept of these 2 instructions...**

/* The below given Assembly program of 8086 Works well.

I can Understand the meaning of every instruction in this code,

except the highlighted 3 instructions. */

The code accepts the input till 0 is hit!

```
Code:
    ASSUME CS:CODE
    CODE SEGMENT
    ORG 1000H
    MOV AX,CS
    MOV DS,AX
BACK:
   MOV AH,01H
   INT 21H
   CMP AL,'0'
   JZ LAST
   JMP BACK
LAST:
   MOV AX,4C00H
   INT 21H
   CODE ENDS

   END
```

assembly 8086

edited Mar 19 '11 at 19:22

asked Mar 19 '11 at 19:12

Muthu Ganapathy Nathar

100 3 14

3 Answers

To really explain the concept, we have to back up to the basic idea of segments, and how the x86 uses them (in real mode).

The 8086 has 20-bit addressing, but only 16-bit registers. To generate 20-bit addresses, it combines a segment with an offset. The segment has to be in a segment register (CS, DS, ES, or SS). You then generate an offset (as an immediate value, or the contents of another register or two.

So, to generate an address, a 16-bit segment register is shifted left four bits, and then a 16-bit offset in some other register is added to that, and the combined total is actually used as the address. Most instructions have a default segment attached to them -- `push`, `pop` and anything relative to `bp` will use `ss`. Jumps and such use `cs`. Some of the string instructions `es` (e.g., `scans`) and some use use two segments -- for example, `movsd` copies data from `[ds:si]` to `[es:di]`. Most other instructions use `ds`. You can also use segment overrides to explicitly specify an address like `es:bx`.

In any case, before you can make any meaningful use of a segment register, you first have to load it with the (top 16 bits of) the address of the data you care about. A typical "small model" program will start with something like:

```
mov ax, @Data
mov ds, ax
```

In tiny model, you use the same segment for the data and the code. To make sure it's referring to the correct segment, you want to get the 16 bits from CS and copy it to DS. As a number of others have mentioned, there's no instruction to move CS directly to DS. The question mentions one possibility; another common one is:

```
push cs
pop ds
```

answered Mar 19 '11 at 20:54

Jerry Coffin
198k 13 158 415

ORG 1000H tells the assembler that the code that follows should be placed at offset 1000H in the code image.

The other two instructions copy CS to DS. It is not copying the segment itself, just updating the pointer to the data segment. For a small program (<64K), static data (string literals in the source, indirect jump tables) may be placed together in the same segment with the code. The segment base pointer needs to be loaded in DS before accessing the static data. The loader (the part of the OS that reads the program from disk to memory and starts it running) has to set CS so that it can run the program, but may not set DS, so the program copies CS to DS when it starts.

The two instruction sequence is needed because "MOV DS, CS" is not a legal 8086 instruction.

answered Mar 19 '11 at 19:24

Andy
3,144 7 13

Ya. I am a newbie to assembly. Can you explain the things in Detail.(If you didnt mind) – Muthu Ganapathy Nathan Mar 19 '11 at 19:32

4 @Muthu: sounds like you ought to just read some standard resources on the subject. SO answers are great for answering specific well-defined questions, but not if you need *everything* explained in detail. Don't ask others to do your studying for you – jalf Mar 19 '11 at 20:31

you can't do

```
MOV DS, CS
```

it's an invalid operation (masm 32: `error A2070: invalid instruction operands`).

```
MOV AX, CS
MOV DS, AX
```

These 2 instruction perform the same as `mov ds, cs` (which is invalid). This way the assembler is happy and doesn't complain. But I can't tell you why the programmer wants the data segment to be the same as the code segment

edited Jun 2 '11 at 16:51

Brian Knoblauch
7,365 2 26 42

answered Mar 19 '11 at 19:15

BlackBear
8,069 2 15 39

Ya, can you explain me about the Meaning of MOV AX,CS and MOV DS,AX – Muthu Ganapathy Nathan Mar 19 '11 at 19:17

I can understand MOV DS,CS is NotAllowed. But can you tell me about the concept of MOV AX,CS – Muthu Ganapathy Nathan Mar 19 '11 at 19:20

Oh! But is there any answer for that... – Muthu Ganapathy Nathan Mar 19 '11 at 19:23

2 @Muthu: The "concept" is simply "copy the value of the `cs` register` into the `ds` register". That's all `mov` does. Which part of it is not clear? – jalf Mar 19 '11 at 20:33